Send a request

With a successful custom endpoint deployed, you are nearly ready to make a job request. Let's start by constructing our request body that will be submitted.

JSON Request Body

Requests to your endpoint must be made via a JSON formatted request; at a minimum, it will need to have an `input` key containing a dictionary of inputs required to complete your job request. For example, if your handler required an input prompt, you might send in something like this:

```
{
  "input": {
    "prompt": "The lazy brown fox jumps over the"
  }
}
```

Optional Inputs

Along with an input key, other top-level inputs can also be included and offer different functions. If a key is passed in at the top level and not included here, it will be discarded and unavailable to your handler.

> 👍 These optional inputs are available to all endpoints regardless of the worker.

⚓ | Webhook

To be notified of completed jobs, a URL can be passed within the top level of the reque

Ask AI

```
{
  "input": {},
  "webhook": "https://URL.TO.YOUR.WEBHOOK"
}
```

Your webhook endpoint should respond with a 200 status to acknowledge the successful call. If not received, the webhook will be attempted up to 2 times after a 10-second timeout.

A POST request will be sent to your URL when the job is complete. This request will contain the same information you would receive if you fetched the results from the `/status/{job_id}` endpoint.

📜 | Execution Policy

By default, if a job remains `IN_PROGRESS` for longer than 24 hours, the worker that has that job will be terminated. If you know the upper limit (with some margin) on how long a job should take, you can prevent stale or dead workers from running excessively by setting the execution timeout policy.

Setting this policy will limit how long a worker remains `IN_PROGRESS` before the worker is killed.

```
{
  "input": {},
        "policy":{
        "executionTimeout": int # time in milliseconds
        }
}
```

Minimum: 5000ms
Default: 86400000ms (24 hours)

💾 | S3-Compatible Storage

The credentials for S3-compatible object storage can be passed in with the request as follows:

```
{
  "input": {},
  "s3Config": {
    "accessId": "key_id_or_username",
```

Ask AI

```
    "accessSecret": "key_secret_or_password",
    "bucketName": "storage_location_name",
    "endpointUrl": "storage_location_address"
  }
}
```

The configuration is only passed onto the worker; it will not be returned as part of the job request output.

Note: The serverless worker will need to contain the logic/functionality that allows it to make sure of this input. If you build a custom endpoint and request s3Config in the input, your worker is ultimately responsible for using the information passed in to upload the output.

Edit this page

Previous
« Get started

Next
Manage Endpoints »

Ask AI